FACULTAD
DE INGENIERIA
Universidad de Buenos Aires

CARRERA DE ESPECIALIZACIÓN EN
SISTEMAS EMBEBIDOS

MEMORIA DEL TRABAJO FINAL

Realidad Aumentada para la optimización de procedimientos industriales

Autor:
Iván Szkrabko

Director:
Leandro Lanzieri (UTN)

Jurados:
Daniel Marquez (pertenencia)
Roberto Compañy (pertenencia)
Matías Álvarez (pertenencia)

Este trabajo fue realizado en la Ciudad Autónoma de Buenos Aires, entre Junio de 2020 y Junio de 2021.

Resumen

En este trabajo se presenta el desarrollo de una aplicación de realidad aumentada para uso industrial. El trabajo surge como una prueba de concepto para la empresa ABB. La aplicación, se desarrolló con el fin de mejorar el desempeño de los operadores en la ejecución de procedimientos industriales, típicos en plantas de lubricantes o farmacéuticas, donde la producción tiene que cumplir con un estricto control de calidad.
Para la ejecución del trabajo se utilizaron conceptos de programación orientada a objetos, testing y programación multihilos. También, se aplicaron conocimientos referentes al desarrollo de endpoints para la API y gestión de base de datos. Por otro lado, gracias a la gestión de proyectos, se logro una buena comunicación con los interesados y llevar el control a lo largo de todo el desarrollo.

Agradecimientos

Agradezco a los jurados y al director de la tesis por su dedicación a la investigación y el desarrollo.

Índice general

Índice de figuras

Índice de tablas

Dedicado a mi familia

Capítulo 1

Introducción general

En este capítulo se realiza una introducción a la realidad aumentada y los sistemas de control distribuidos. Hacia el final, se explica la motivación, alcance y objetivos del presente trabajo.

1.1. Realidad Aumentada

1.1.1. Tecnología

La digitalización del mundo real tiene distintos niveles. Podemos clasificarla en realidad virtual, realidad aumentada y realidad mixta, como puede verse en la figura 1.1:

FIGURA 1.1. Clasificación de tecnologías[1].

La realidad virtual, consiste en un ambiente completamente digital. Se buscan aislar los sentidos del usuario del mundo real. De esta manera el usuario queda inmerso en otra realidad con la cual puede interactuar. Los dispositivos que permiten esta interacción son los cascos de realidad virtual y sus *joysticks*.

La realidad aumentada es una tecnología que permite conectar los dos mundos, el mundo físico y el mundo digital. Esto se logra superponiendo imágenes sobre la percepción que tenemos del mundo real, mediante el uso de hologramas o incluso con teléfonos celulares. Donde al activar la cámara, nos muestra en la pantalla no solo el ambiente en el que estamos, sino que vemos proyectada información adicional. Ejemplos de esta tecnología, son los populares filtros de Instagram o el juego PokemonGo.

[1]Imagen tomada de https://scand.com/company/blog/augmented-mixed-and-virtual-reality-for-business/

Por ultimo, la realidad mixta. Esta tecnología es una mejora de la realidad aumentada. Los elementos no solo son adicionados sobre la percepción visual, sino que interactúan ademas con el espacio físico. Por ejemplo, si tuviéramos una pelota en una aplicación de realidad mixta, la misma podría rodar cuesta abajo de una escalera.

1.1.2. Procedimientos *batch*

Una operación *batch* o por lotes es aquella en la cual las condiciones de operación cambian con el tiempo. Por lo general, se siguen una serie de pasos, el producto se transfiere a la siguiente fase de la operación, y el proceso comienza de nuevo. En la figura 1.2 podemos ver que en un proceso por lotes, solo cuando llegamos a t=4, obtenemos el producto final deseado:

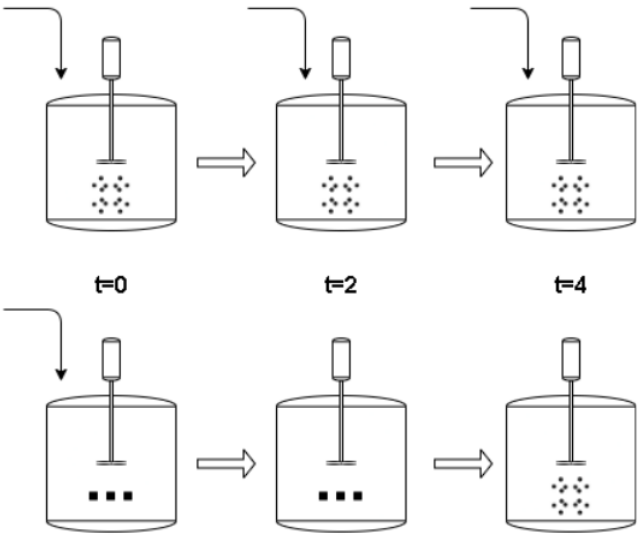

FIGURA 1.2. proceso continuo (superior) vs. proceso por lotes (inferior)[2].

Los procedimientos operativos para los procesos por lotes son generalmente más complicados y difíciles de escribir que para las instalaciones de estado estacionario porque el tiempo es un factor. Además, muchas instalaciones están diseñadas para fabricar varios productos a partir de los mismos equipamientos en diferentes momentos. Por lo tanto, se necesitan dos o más procedimientos para los mismos equipos de la planta, creando la posibilidad de confusiones y malentendidos.

Algunas operaciones por lotes implican el uso de hojas de cálculo. Por ejemplo, un operador puede agregar una bolsa de químicos a un reactor y luego agregar un segundo químico. La razón del segundo al primero debe ser exacta. El operador pesará la primera bolsa, calculará el peso del segundo químico necesario y procederá a pesar ese químico. Se necesita entonces una hoja de cálculo para determinar los requisitos para el peso del segundo producto.

1.1.3. Sistemas de control distribuidos

Los sistema de control distribuidos, también conocidos por sus siglas en ingles DCS. Son sistemas compuestos por sensores, controladores y computadoras que se distribuyen por toda la planta. Cada uno de estos elementos tiene un propósito único, como la adquisición de datos, el control de procesos, el almacenamiento de datos o la visualización gráfica. Estos elementos individuales se comunican con una sistema centralizado a través de la red de área local de la planta, denominada red de control. A continuación se muestra en la figura 1.3, la arquitectura de red del sistema DCS de ABB, usado en este trabajo:

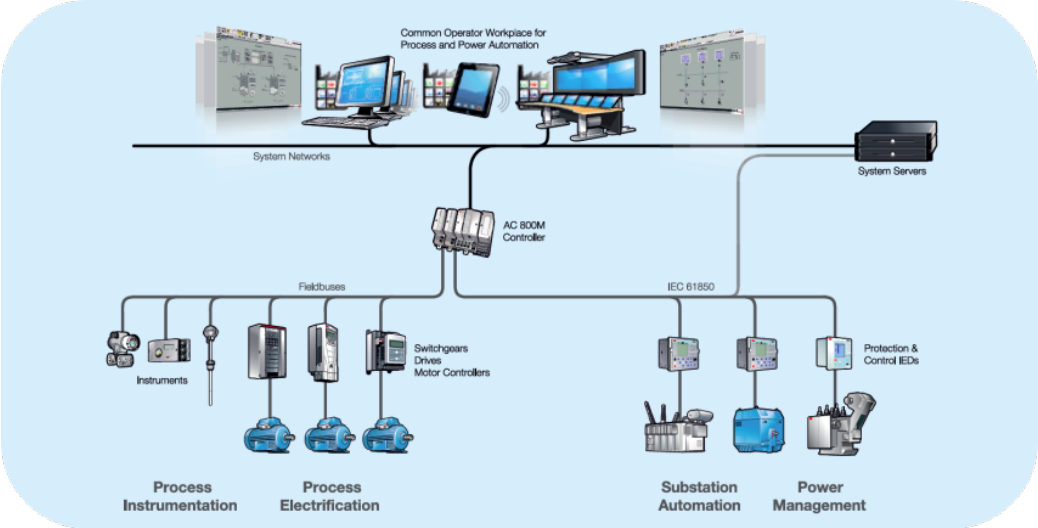

FIGURA 1.3. Sistema de control distribuido 800xA[3].

El DCS toma decisiones automatizadas basadas en las tendencias de producción de toda la planta. Como ejemplo, el DCS de una planta de energía, podría aumentar automáticamente la capacidad de generación de vapor de múltiples turbinas, para adecuarse a la demanda cambiante de electricidad. Mientras que un PLC puede ajustar la operación de una sola unidad, el DCS puede realizar ajustes en cada una de las unidades de control que interactúan en una planta.

1.1.4. Protocolo OPC

OPC (*Open Platform Communications*) es el estándar de interoperatividad para el intercambio seguro y confiable de datos en la industria. Tiene especial uso en los DCS, dado que son sistemas compuestos por productos de distintos rubros y fabricantes. El protocolo es independiente de la plataforma y garantiza el flujo continuo de información entre dispositivos de múltiples proveedores. La Fundación OPC es responsable del desarrollo y mantenimiento de este estándar. Estas especificaciones definen el acceso a datos en tiempo real, monitoreo de alarmas y eventos, acceso a datos históricos y otras aplicaciones.

[3]Imagen tomada de https://new.abb.com/control-systems/system-800xa/800xa-dcs/system/architecture

1.1.5. Otros trabajos

A continuación se presenta un listado de algunos trabajos contemporáneos relacionados con la realidad aumentada y su aplicación en la industria.

Smart retrofitting in the context of industry 4.0

En este trabajo publicado en Elsevier (CIRP 88 369–374), se utiliza el Hololens para capacitar a los operadores luego de una mejora en una maquina dobladora de caños. El trabajo destaca que la herramienta permite a los operadores adaptarse rápidamente a la maquina, permitiéndoles trabajar de manera segura y guiada.

Production workplace enhanced by mixed reality

En este paper presentado en la conferencia *Mensch und Computer* 2020 en Alemania (muc2020-ws116-005), se muestra una aplicación de realidad aumentada para guiar al operador en el ensamblado de un scooter. El trabajo destaca el análisis de la relación costo-beneficio a la hora de armar procedimientos para los operadores. Plantea que deben evaluarse, el tiempo y presupuesto para el desarrollo del procedimiento, en comparación con el costo de una falla de ensamblado.

Mixed reality technology for onsite construction assembly

En este paper presentado en la conferencia *Materials Science, Engineering and Chemistry* 2020 (Matec 312,06001), se presenta una aplicación de realidad aumentada cuyo objetivos es comunicar los detalles de diseño para el montaje y la construcción. El trabajo destaca que el Hololens es capaz de guiar la instalación de componentes eléctricos y de plomería, logrando un 9 % de mejora en la productividad. Menciona ademas, que las imágenes ocluidas en condiciones de luz solar, son el desafió a superar por el HoloLens antes de una adopción generalizada.

1.2. Motivación

El propósito de este proyecto es innovar en la interacción entre los operadores y los sistema de control distribuidos, para impulsar nuevas soluciones en el área de la automatización industrial. Se busca explorar las oportunidades que ofrece la realidad aumentada para mejorar y optimizar las tareas de los operadores, además de agilizar el entrenamiento de nuevos operarios y mejorar la seguridad para procedimientos bajo situaciones de emergencia.

1.3. Alcance

En la presente memoria, se documenta el trabajo realizado donde se incluyen los siguientes aspectos:

- Desarrollo de la aplicación de realidad aumentada.

- Desarrollo del conector OPC.
- Desarrollo de API REST.
- Comunicación entre el DCS y la interfaz holográfica.
- Implementación de base de datos cloud.

Capítulo 2

Introducción específica

En este capítulo se presentan las herramientas usadas para desarrollar el trabajo y un breve compendio de otros trabajos similares en la temática. Hacia el final, se encuentran los requerimientos y la planificación utilizada.

2.1. Entorno de desarrollo

2.1.1. Hololens2

El equipo que se utilizo para la interfaz de realidad aumentada es el Microsoft Hololens 2. Podemos ver una foto del equipo en la figura 2.1:

FIGURA 2.1. hololens2[1].

Fue lanzado al mercado para uso corporativo, Microsoft apuesta a que la tecnología sea usada con fines productivos en distintas industrias. Es un equipo de ultima generación que funciona con un sistema operativo Windows 10 core. La tabla 2.1, muestra sus especificaciones técnicas:

Lo innovador de su diseño es que utiliza micro espejos para proyectar 3 haces de luz láser de color rojo, verde y azul. Con un micro espejo por cada ojo oscilando a 12.000 veces por segundo, compone la imagen horizontal. Con un segundo micro espejo, oscilando a 120 veces por segundo forma la componente vertical.

[1]Imagen tomada de https://img-prod-cms-rt-microsoft-com.akamaized.net/cms/api/am/imageFileData/

TABLA 2.1. Especificaciones técnicas

Hardware	Descripción
Procesador	Qualcomm Snapdragon 850
Memoria	4 GB DRAM
Almacenamiento	64 GB
Cámara	1080p
Peso	566 gr
CPU holografica	Segunda generación
Resolución	2k 3:2
Unidad de medida inercial	Acelerómetro, giroscopio y magnetómetro
Seguimiento de cabeza	4 cámaras de luz visible
Seguimiento de ojos	2 cámaras de infrarrojas

Luego,este haz de luz es proyectado hacia nuestra retina, utilizando el visor del Hololens 2 que actúa como una guía de onda. En la figura 2.2 podemos ver una ejemplificación del funcionamiento interno:

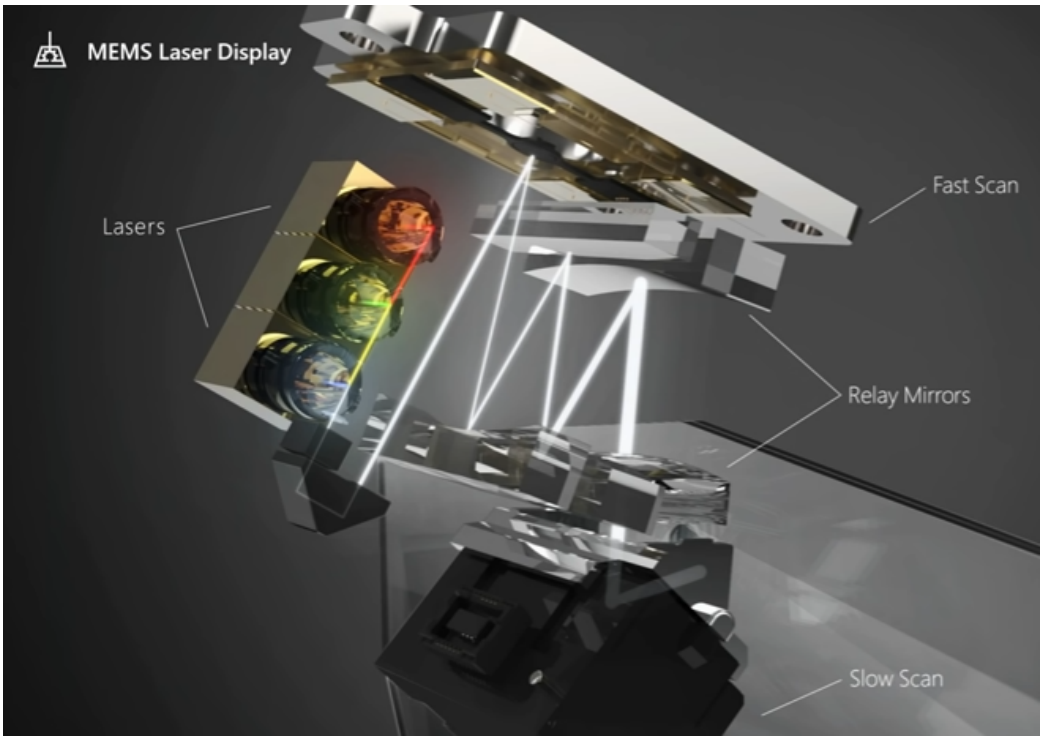

FIGURA 2.2. micro espejos[2].

2.1.2. MRTK

MRTK(*Mixed Reality Toolkit*) para Unity es un kit de desarrollo multiplataforma de código abierto para aplicaciones de realidad mixta. Proporciona un sistema de entrada multiplataforma, componentes fundamentales y bloques de construcción comunes para interacciones espaciales. Tiene como objetivo acelerar el desarrollo de aplicaciones para Microsoft HoloLens, los visores inmersivos de Windows

[2]Imagen tomada de https://img-prod-cms-rt-microsoft-com.akamaized.net/cms/api/am/imageFileData/

Mixed Reality y la plataforma OpenVR. El proyecto busca reducir las barreras de entrada, crear aplicaciones de realidad mixta y contribuir a la comunidad. El *framework* puede desglosarse conceptualmente como se muestra en la 2.3:

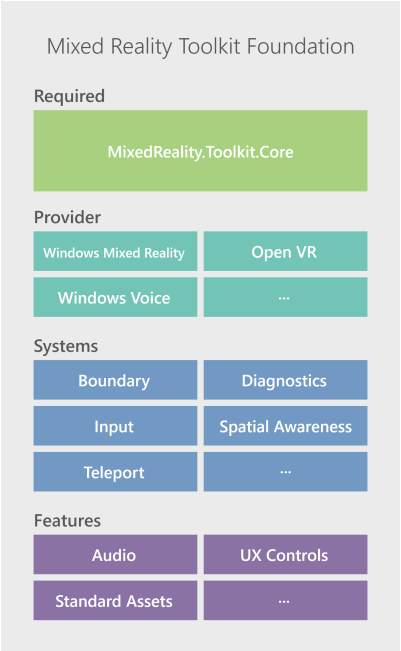

FIGURA 2.3. componentes del *framework* [3].

Sus componentes principales son:

- *Hands*: clase básica de soporte con servicios para seguimiento de las manos.
- *ObjectMeshObserver*: procesamiento del ambiente usando el modelado 3D.
- *WindowsMixedReality*:compatibilidad con dispositivos *Windows Mixed Reality*, incluidos Microsoft HoloLens y auriculares inmersivos.
- *Profiles*: perfiles predeterminados para los sistemas y servicios de *Microsoft Mixed Reality Toolkit*.
- *SceneSystem*: sistema que proporciona compatibilidad con aplicaciones de múltiples escenas..
- *StandardAssets*: renderizado estándar, materiales básicos y otros activos para experiencias de realidad mixta

Ademas del MRTK, se utilizan programas de diseño como el Fusion360 para crear objetos 3D. Se utiliza el Visual Studio como IDE de desarrollo y como herramienta de compilación. Y finalmente Unity, como herramienta de diseño visual de la aplicación. En la figura 2.4, podemos ver un diagrama del entorno completo:

[3]Imagen tomada de https://docs.microsoft.com/en-us/windows/mixed-reality/

[4]Imagen tomada de https://img-prod-cms-rt-microsoft-com.akamaized.net/cms/api/am/imageFileData/

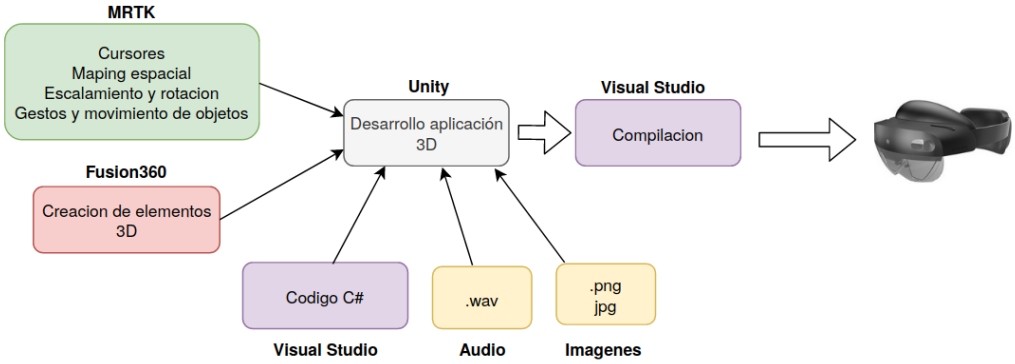

FIGURA 2.4. entorno de desarrollo[4].

2.1.3. OPC *framework*

Las especificaciones de OPC *Classic* se basan en la tecnología de Microsoft Windows, utilizando COM / DCOM (*Distributed Component Object Model*) para la comunicación entre componentes de software en una red distribuida cliente-servidor. La especificación original es OPC-DA (*Data Access*), que define una interfaz entre las aplicaciones cliente y servidor para intercambiar datos de proceso y fabricación. Antes de OPC-DA, los productos de distintos proveedores (PLC, HMI) requerían que cualquier dispositivo o aplicación que se conectara a ellos, tuviera un "controlador personalizado" que comunicara el dispositivo con equipos de terceros. En la figura 2.5 se muestra una arquitectura típica de control previa a la existencia del *standard* OPC:

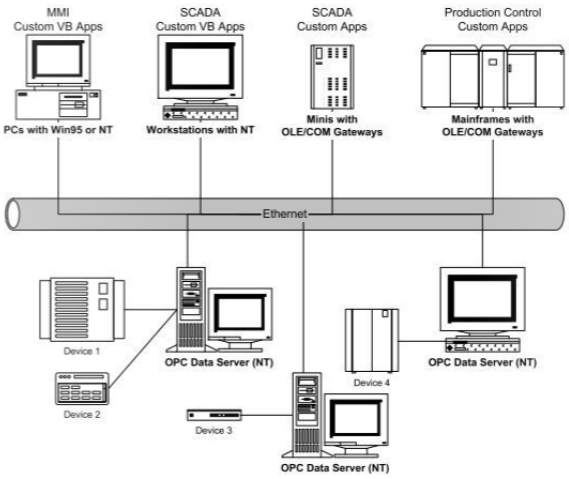

FIGURA 2.5. arquitectura OPC[5].

Hubo muchos problemas asociados con las comunicaciones basadas en controladores personalizados. La tecnología patentada de alto costo que forzaba a los usuarios a mantener un proveedor en particular. Dificultades para configurar y mantener actualizado los *drivers* debido al lanzamiento de nuevos dispositivos y aplicaciones. OPC-DA hizo posible la conexión a cualquier fuente de datos en tiempo real sin un driver escrito específicamente para el par fuente de datos /

[5]Imagen tomada de https://technosoftware.com/opc-daaehda-client-net

receptor de datos. Por lo tanto, las lecturas y escrituras se pueden realizar sin que el receptor de datos tenga que conocer el protocolo nativo de la fuente de datos o la estructura de datos interna.

La especificación describe los objetos y sus interfaces, las cuales son implementadas por los servidores OPC. Aunque OPC está diseñado principalmente para acceder a datos desde un servidor en red, las interfaces se pueden utilizar en muchos lugares dentro de una aplicación. En el nivel más bajo, pueden obtener datos sin procesar de los dispositivos físicos y enviarlos a un SCADA o DCS, o desde el sistema SCADA o DCS a la aplicación. La arquitectura y el diseño permiten que una aplicación cliente acceda a datos de muchos servidores OPC proporcionados por distintos proveedores. En la figura 2.6 se muestra un diagrama en bloques para ejemplificar lo explicado:

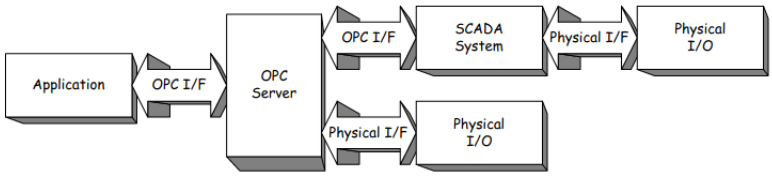

FIGURA 2.6. diagrama en bloques[6].

A alto nivel, un servidor OPC se compone de varios objetos: el servidor, el grupo y el elemento. El objeto del servidor OPC mantiene información sobre el servidor y sirve como contenedor para los objetos del grupo. El objeto del grupo mantiene información sobre sí mismo y proporciona el mecanismo para contener y organizar lógicamente los elementos OPC. Un cliente se comunica con el servidor a través de las interfaces para cada objeto OPC, como se ve en la figura 2.7:

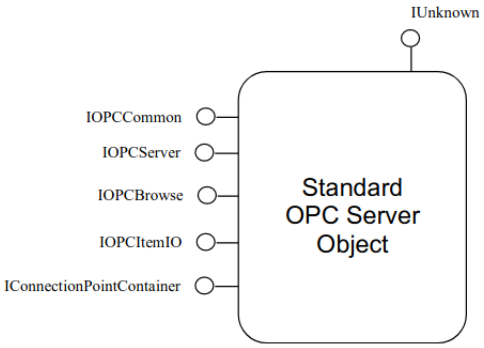

FIGURA 2.7. diagrama en bloques[7].

- IOPCServer:ésta es la interfaz principal de un servidor OPC, permite realizar las consultar a los grupos de elementos OPC.
- IUnknown: consiste en un puntero a una tabla que permite descubrir dinámicamente si un objeto admite una interfaz en particular.
- IOPCCommon:Proporciona la capacidad de configurar y consultar el LocaleID para la sesión cliente/servidor en particular. El LocaleID contempla el

[6]Imagen tomada de https://opcfoundation.org

[7]Imagen tomada de https://opcfoundation.org

formato de números, fechas, moneda, etc. puede depender de la configuración regional.

- IOPCBrowse: proporciona métodos mejorados para navegar por el espacio de direcciones del servidor y para obtener propiedades de los elementos OPC.
- IConnectionPointContainer: permite disparar eventos llamando a un método de la interfaz de un objeto COM del lado del cliente implementado por el mismo cliente.
- IOPCItemIO: en términos de rendimiento esta interfaz se comportará como si se creara un grupo, se agregaran los elementos OPC, se realizara una sola lectura o escritura y luego se eliminara el grupo. Por lo tanto no es la interfaz recomendada para la mayoría de las aplicaciones.

Los grupos OPC proporcionan una forma para que los clientes organicen los datos. Por ejemplo, el grupo puede representar elementos en una pantalla de operación o variables de un lazo PID. Los datos se pueden leer y escribir. Los elementos OPC representan conexiones a fuentes de datos dentro del servidor. Un elemento OPC, desde la perspectiva de la interfaz, no es accesible como objeto por un cliente OPC. Por lo tanto, no hay una interfaz externa definida para un elemento OPC. Todo el acceso a los elementos OPC se realiza a través de un objeto de grupo que "contiene" los elementos. Un grupo también proporciona una forma para que el cliente se "suscriba" a la lista de elementos para que pueda ser notificado cuando cambien. Este ratio de actualización es un parámetro configurable del servidor.

2.1.4. Requerimientos

A continuación se listan los requerimientos en base a las distintas etapas del trabajo:

1. Requerimientos asociados al desarrollo de la interfaz visual:
   a) La interfaz debe ser intuitiva y simple.
   b) El idioma definido es Español.
2. Requerimientos asociados al desarrollo de lógica en .NET:
   a) La aplicación debe ser fluida y responder sin demoras apreciables por el operador, estableciéndose así el límite máximo de espera en 2 segundos.
   b) La aplicación debe poder hacer operaciones GET y POST sobre un servidor web, ya sea local o en la nube.
3. Requerimientos asociados a la API rest:
   a) La API no será de acceso público, solo podrá ser consultada por las aplicaciones que poseen un *token* de seguridad.
4. Requerimientos asociados a la interfaz de comunicación con el sistema de control:
   a) La solución debe poder consultar una serie de datos específicos a elección, de los elementos que pertenecen al sistema de control.

b) La comunicación debe cumplir con el *standard* OPC.

2.1.5. Planificación

El trabajo fue dividido en tareas y se estimaron los tiempos que debían emplearse para cada una de ellas. A su vez, se analizaron qué tareas debían realizarse primero y cuáles eran sus dependencias. Al momento de organizar las tareas se consideraron 15 horas semanales efectivas de trabajo, obteniendo como resultado la planificación de la figura 2.8:

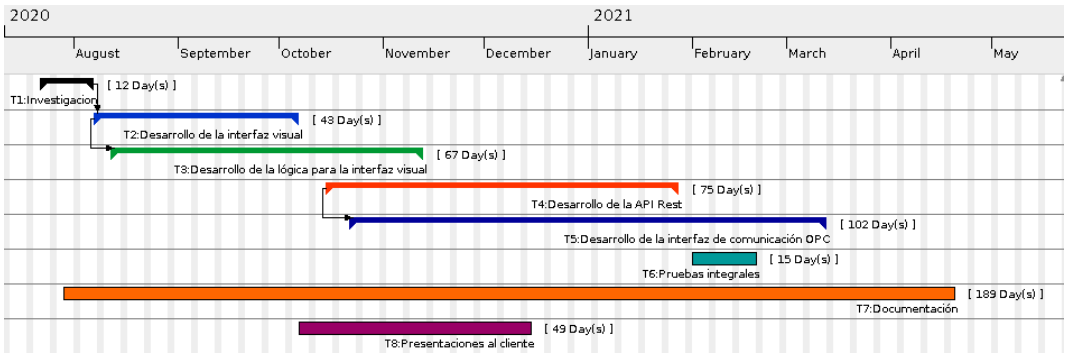

FIGURA 2.8. diagrama de Gantt[8].

Como se estimo al inicio del trabajo la comunicación OPC fue la que mas tiempo empleo, incluso se extendió 1 mes mas de lo planeado.

Capítulo 3

Diseño e implementación

3.1. Análisis del software

3.1.1. Arquitectura del sistema

La solución esta integrada por 6 componentes principales. La interfaz visual, diseñada en Unity3D, a través de la cual el usuario interactúa con la aplicación. El backend de esta interfaz, desarrollado en .NET utilizando CSharp como lenguaje de programación. La APIrest, a través de la cual la aplicación embebida en el Hololens2 se comunica con el servidor web. La base de datos, que es actualizada a través de la APIrest. El servicio OPC, desde el cual se enviarán los datos pertinentes al sistema de control, a través del *standard* industrial OPC. Y finalmente el sistema de control, representado por un controlador simulado. En la Figura 3.1 se puede observar el diagrama en bloques:

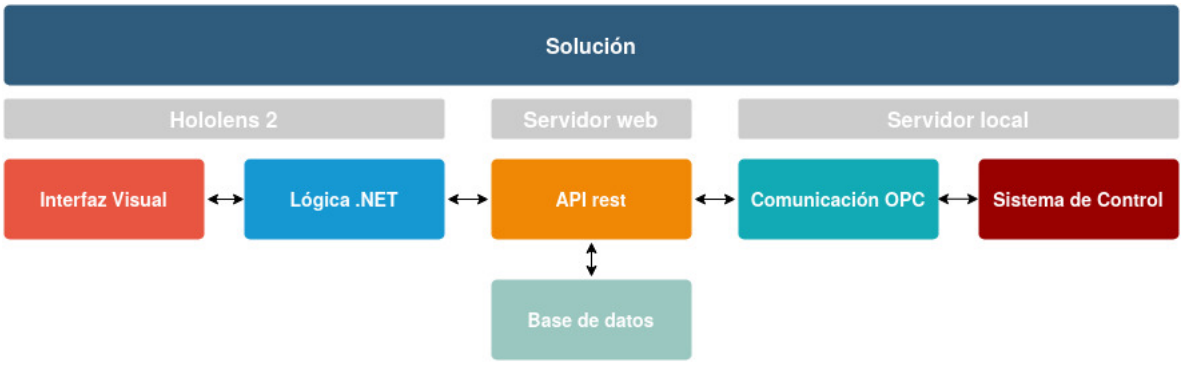

FIGURA 3.1. diagrama en bloques[1].

Imagen de elementos en la DB Imagen de lo devuelto por la api Imagen SFC control

3.1.2. Interfaz de realidad aumentada

Aplicación

La aplicación se encarga de ayudar a los operadores durante la fabricación por lotes de un producto determinado. Durante los procedimientos *batch* se utilizan el mismo conjunto de maquinas para la fabricación de distintos productos. Por lo

tanto, las tareas de los operadores varían en las distintas partes del proceso dependiendo del producto que se este fabricando. Una etapa clave en la fabricación del producto, es el agregado de aditivos en la mezcla. La aplicación desarrollada se encarga de evitar que el operador falle a la hora de seleccionar y dosificar el aditivo requerido. La aplicación comienza con la siguiente imagen: producir

En este punto se aguarda la lectura del codigo QR correspondiente a la receta a producir. Las recetas son el conjunto de parámetros que involucran un lote de producción. En ellas podemos encontrar el listado de aditivos, la cantidades requeridas, los códigos de los distintos productos, tiempos de mezclado y demás parámetros necesarios para la producción. Si el operador lee un código de receta para el cual, el sistema de control no fue configurado a producir en ese momento, se indicara un error como muestra la siguiente figura:

Para poder avanzar, es necesario que se lea el código de la receta correcta. Como podemos ver a continuación:

En este punto el operador acepta la receta y la interfaz mostrara la siguiente imagen:

En el margen derecho podemos ver 3 círculos azules, los cuales representan 3 tareas del operador a lo largo de la receta de producción. En la parte superior del listado de tareas podemos leer el producto que se fabricara en esta receta. En el lado izquierdo superior podemos ver una leyenda que ira actualizándose para guiar al operador. En la parte inferior encontramos 4 botones con las siguientes funcionalidades:

- Ocultar: permite simplificar la vista holográfica, reduciendo la información proyectada en el hololens.
- Mostrar: restablece la información holográfica oculta.
- Datos: restablece la información correspondiente a la lecrestablece la información holográfica oculta.tura de codigos QR.
- Siguiente: se utiliza para avanzar a la siguiente etapa drestablece la información holográfica oculta.e producción.

Ademas en el centro izquierdo de la imagen, podemos ver datos que hacen referencia a los distintos aditivos que se usaran en la receta. Se muestran los siguientes datos:

- QR: código QR leído por el hololens.
- Código: código del producto leído .
- Producto: nombre del producto leído.
- Cantidad: cantidad necesaria del producto en la receta.

Según lo indicado en pantalla el operador deberá pulsar “Siguiente” para iniciar el procedimiento. En ese momento la instrucción se modificara, solicitándole que vierta en el tanque de producción TK212, el producto A. Como podemos ver en la siguiente imagen:

El operador deberá localizar las bolsas de este producto y leerá con el Hololens el código QR de la misma. En ese momento el conjunto de datos de actualizara como vemos en la siguiente figura:

El operador pulsara “Aceptar” y luego deberá ingresar las cantidades indicadas en el tanque de producción. Luego pulsara “Siguiente” y se actualizara la información con una nueva instrucción como vemos a continuación:

Si el operador hubiese cometido un error al tomar la bolsa de un producto no requerido, la siguiente imagen hubiese alertado al operador, y no hubiese podido avanzar la receta de ninguna manera:

El procedimiento se repetirá para el total de las tareas de la receta. Hasta la ultima operación, luego se mostrara la siguiente imagen indicándole al operador que puede cerrar la aplicación:

Los comandos al sistema de control son enviados luego de que el correcto producto fue leído y el operador avanza al siguiente paso de la receta, garantizando que se cumplió la tarea asignada. Si el producto erróneo es leído, el operador no tiene habilitado el paso siguiente, por lo tanto queda retenido en esa etapa. Ademas, las recetas validas son solo aquellas en las que el sistema de control aguarda operaciones manuales, de esta manera nos aseguramos que no puedan mezclarse otras recetas con la receta activa.

API y base de datos

La API (*Application Programming Interface*) fue desarrollada en .NET y se implementaron las siguientes operaciones:

- GET:
- POST:
- PATCH:
- DELETE:

Se creo una clase “Receta” la cual contiene los siguientes parámetros:

- 1:
- 2:
- 3:

Esta clase también se implemento en la lógica del Hololens. De esta manera, se utilizan los metodos GET y PATCH durante los pasos de la interfaz para adquirir o enviar datos a la API. Cuando desde la interfaz leemos un código QR, internamente estamos enviando una petición GET al servidor web, para traer los datos de la receta y almacenarlos en una instancia de la clase Receta. Si la receta es correcta, utilizaremos el mismo ID de receta, para enviar las actualizaciones de estado cuando el operador avance una etapa. Esta actualización se hace con el verbo PATCH. Este se implemento para modificar solo los parámetros del input de usuario en la receta y no alterar el resto.

La API se hosteo en Azure, la plataforma *cloud* de Microsoft. Se utilizo el servicio Azure Web Apps, que es una plataforma que permite publicar aplicaciones web que se ejecutan en múltiples *frameworks* y escritas en diferentes lenguajes de programación. Esto permitió aumentar la velocidad de desarrollo de la aplicación

dado que la sincronización y el *deployment* son prácticamente instantáneos. Ademas, ofrece múltiples posibilidades de integración con otros servicios a futuro.

El motor de la base de datos utilizado es MongoDB. Es una base de datos NoSQL basada en documentos. Se opto por esta tecnología dado que los parámetros de las recetas podían ser encapsulados en un JSON . Se utilizo la plataforma SaaS de MongoDB, Atlas, para poder brindar flexibilidad y escalabilidad a la solución con una base de datos cloud. Para acceder a la misma se debe utilizar una *key* que solo posee la APIrest. A continuación se muestra la colección de datos creada para la aplicación:

Imagen de mongodb

3.1.3. Sistema de control

Nodo 800xA

Para el sistema de control se utilizo la herramienta de ABB denominada *SoftController*. Esta nos permite simular la presencia de un controlador físico conectado a la red, con la IP del localhost. Para la lógica de control se uso el CBM (*Compact Control Builder*) , que nos permite crear lógicas del mismo tipo que las usadas en los sistemas DCS 800xA de ABB. De hecho las mismas podrían exportarse al sistema 800xA sin problemas. A continuación podemos ver una imagen del *SoftController* simulando un controlador:

En el CBM podemos ver los bloques de las distintas recetas en la aplicación de control, como se muestra en esta figura:

A medida que nuevas recetas sean necesarias, estos bloques funcionan como objetos de una clase que pueden instanciarse para cada tipo de receta. Simultáneamente estas recetas se cargan en la base de datos para establecer la comunicación entre el sistema de control y la interfaz del Hololens. Dentro de la receta podemos ver una lógica secuencial del procedimiento, que ira avanzando a medida que el operador complete los pasos en la interfaz holográfica. Podemos ver una imagen del avance de la secuencia en el tiempo:

Por ultimo tenemos el OPC server. Esta aplicación crea un OPC server con acceso a las variables del sistema de control. Para conectarnos debemos apuntar a la IP del controlador simulado como se muestra a continuación:

Este nodo se implemento en una maquina virtual usando VirtualBox, posee un sistema operativo Windows 10 con 4 gb de memoria RAM y 60 Gb de disco. La ventaja radica en la portabilidad del nodo, dado que podría integrarse a un servidor ESXI en cualquier entorno industrial.

Servicio de actualización

El servicio se instalo en el mismo nodo virtualizado 800xA visto anteriormente. Se encarga de actualizar la base de datos según los cambios en el sistema de control. Y viceversa, también es capaz de actualizar las variables de control de acuerdo a la base de datos. Se desarrollo en .NET y se ejecuta como un servicio mas de

Windows. El servicio incluye el *framework* OPC mencionado en el Capitulo 2, lo que le permite leer y escribir variables OPC.

Una vez en ejecución se encarga de realizar consultas GET a la APIrest de cada una de las recetas en ejecución. De esta manera es capaz de detectar que los insumos ya fueron vertidos en el tanque y avanzar así a la siguiente etapa. Ademas, se encarga de setear la variable "waitinput", la misma es clave en el funcionamiento de la aplicación dado que indica que la receta se encuentra en ejecución y aguarda ordenes del operador. De esta manera se valida que el operador solo pueda iniciar la aplicación con la receta en ejecución por mas que intente leer otros códigos de receta inválidos.

A continuación podemos ver el código de la implementación de los métodos OPC:

```
#define MAX_SENSOR_NUMBER 3
#define MAX_ALARM_NUMBER  6
#define MAX_ACTUATOR_NUMBER 6

uint32_t sensorValue[MAX_SENSOR_NUMBER];
FunctionalState alarmControl[MAX_ALARM_NUMBER]; //ENABLE or DISABLE
state_t alarmState[MAX_ALARM_NUMBER];           //ON or OFF
state_t actuatorState[MAX_ACTUATOR_NUMBER];     //ON or OFF

void vControl() {

  initGlobalVariables();

  period = 500 ms;

  while(1) {

    ticks = xTaskGetTickCount();

    updateSensors();

    updateAlarms();

    controlActuators();

    vTaskDelayUntil(&ticks, period);
  }
}
```

CÓDIGO 3.1. Pseudocódigo del lazo principal de control.

Capítulo 4

Ensayos y resultados

4.1. Evaluación de las interfaces

A continuación podemos ver algunas mediciones de tiempos de respuesta de las interfaces principales. Para la medición Hololens - API, se utilizo la opción *debug* del Hololens para identificar el *timestamp* del comando emitido al clickear "Siguiente" en la interfaz. Ademas se utilizo la API para loguear el *timestamp* en el que se recibe la consulta web. Para la medición API - server OPC, también se utilizo la API para loguear el *timestamp* en el que se recibe la consulta web. Y se utilizo el servicio OPC para loguear el momento en el que la variable OPC es actualizada. Para un promedio de 10 mediciones consecutivas, los resultados pueden verse en la tabla 4.1:

TABLA 4.1. Tiempos de respuesta promedio

Interfaz	**Tiempo**
Hololens - API	>340ms
API - server OPC	>320ms

Dado que la aplicación se comunica a través de mensajes del tipo JSON y la cantidad de información enviada es reducida, los tiempos de respuesta son realmente bajos y no representan un problema para la implementación. Podemos ver que el tiempo de respuesta de toda la cadena de comunicación, es inferior al segundo. Este trabajo se realizo utilizando servidores *cloud* pero bien podrían ser locales. Llegado el caso, si los tiempos representan una perdida de fluidez en la aplicación, la migración no requeriría mayores esfuerzos.

Capítulo 5

Conclusiones

5.1. Resultados obtenidos

Fue muy importante la planificación inicial para enmarcar y organizar el trabajo. Las estimaciones fueron correctas, aunque se subestimo la etapa de integración de todos los componentes. El trabajo se extendió hasta el mes de Junio solapándose con la redacción de la tesis.

El trabajo respeto los requerimientos propuestos, pero no se logro una robustez en cuanto a la seguridad informática. En principio, estaba pensada la utilización de un *token* de seguridad para la API pero fue desestimado conforme avanzaba el proyecto. Principalmente porque mientras la conexión no sea encriptada la utilización del *token* carece de sentido. Esto es algo que se plantea como una mejora a futuro.

El trabajo logro integrar realidad aumentada, plataformas *cloud* y sistemas de control tradicionales con éxito. La aplicación resulto útil en las pruebas realizadas, y durante las presentaciones con clientes industriales el *feedback* fue positivo. El trabajo demostró que las integraciones de sistemas de control con las tecnologías 4.0 son posibles, pero es necesario encontrar casos de uso prácticos que justifiquen el esfuerzo del desarrollo. La mano de obra no solo puede automatizarse, sino que también puede mejorarse con ayuda de las nuevas tecnologías.

5.2. Próximos pasos

El siguiente paso es utilizar un *token* para autenticar la comunicación y certificados TLS para encriptar el canal de comunicación entre el Hololens y el servidor web. De esta manera se mejoraría la seguridad considerablemente. Utilizando como todo lo desarrollado se tiene un buen punto de partida para realizar una segunda aplicación orientada al mantenimiento de activos en la planta.